Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/EP05/050693

International filing date: 16 February 2005 (16.02.2005)

Document type: Certified copy of priority document

Document details: Country/Office: EP
Number: 04290417.7
Filing date: 16 February 2004 (16.02.2004)

Date of receipt at the International Bureau: 15 April 2005 (15.04.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

Europäisches Patentamt
European Patent Office
Office européen des brevets

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | Patent application No. | Demande de brevet n°

04290417.7

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

Anmeldung Nr:
Application no.: 04290417.7
Demande no:

Anmeldetag:
Date of filing: 16.02.04
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

Thomson Licensing S.A.
46, quai A.Le Gallo
92100 Boulogne-Billancourt
FRANCE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

Method for inserting a new device in a community of devices

In Anspruch genommene Priorität(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

H04L29/00

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LU MC NL PT RO SE SI SK TR LI

FIELD OF THE INVENTION

The invention applies to digital networks and communities of devices.

BACKGROUND ART

A community of devices is a set of devices linked by mutual trust relations. One can refer to "*Secure Long Term Communities in Ad Hoc Networks, N. Prigent, C. Bidan, J.P. Andreaux, O. Heen, , 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*" to find explanations on communities of devices.

Communities are generally encountered in the following domains: ad-hoc networks, digital home networks, collection of UPnP™ devices, personal area networks, sensor networks, secured wireless networks (e.g. networks according to the IEEE 802.11a standard with WPA).

A community of devices, or simply a community, exists when all devices of a set have set-up mutual (i.e. symmetrical) trust relations with all other devices of this set. Trust relations may be transitive that is, if the situation of Fig. 1a (device a and device b have set-up a mutual trust relation) and Fig. 1b (device b and device c have set-up a mutual trust relation) holds, then the situation of Fig. 1c also holds (i.e. devices a and c mutually trust each other).

An insertion operation happens when a new device has to be inserted in a community. At the end of the insertion operation, the new device belongs to the community. That is, all other community devices have established trust relations with the new device. These relations may be transitive.

Existing device insertion methods require at least one user action, to authorize the entry of the new device in the community.

Fig. 2 illustrates a first known method to obtain user authorization before effective device insertion. This method of authorization using a trusted device is described in "*Frank Stajano and Ross Anderson, The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks (1999), available at the following address: http://citeseer.nj.nec.com/stajano99resurrecting.html*".

In this method, there are four major steps:

In step 2.1, "Insertion request", a trusted device 2 of a community 5 named *Detector* detects an insertion request from a new device 1 named *New*.

In step 2.2, "User request", the *Detector* 2 asks the user 3 for authorization. This step can necessitate user authentication on device 2, using state of the art methods (password, biometry...).

In step 2.3, "User confirm", the user 3 explicitly authorizes the insertion of the new device 1.

In step 2.4, "Insertion confirm", the authorization is sent to the new device 1.

These four main steps can be secured using cryptographic material. They can be followed by other optional steps to exchange additional key material or protocol related information.

It should be noted that this method does not allow the user 3 to choose the device used to authorize the insertion: only the detector device 2 can be used.

Fig. 3 illustrates a second method of authorization using a predetermined controller (direct method) described for example in standard ANSI/IEEE Std 802.11 [ISO/IEC DIS 8802-11] *"Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999"*, available at the following address: *"http://standards.ieee.org/getieee802/download/802.11-1999.pdf"*.

In this method, there are three major steps:

In step 3.1, "User request", the user 3 directly requests a device insertion, using a user interface of the predetermined controller 4 of a community 5.

In step 3.2, "User confirm", the user 3 explicitly authorizes the insertion of the new device 1 in the community 5.

In step 3.3, "Insertion confirm", the authorization is sent to the new device 1.

These three main steps are generally followed by other steps, among which an action from the user on the new device. This method is used in centralized situations and when there exists pre-shared key to insure trust between community devices. This is the case for 802.11 networks with WEP keys and an access point used as a controller.

It is to be noted that this method does not allow the user 3 to choose the device used to authorize the insertion: only the controller device 4 can be used.

Fig. 4 illustrates a third method of authorization using a predetermined controller (indirect method) described in "*Ed Callaway et al.: Home networking with IEEE 802.15.4 : a developing standard for low rate WPAN, IEEE Com. Mag. August 2002, pp. 70-76*".

In this method, there are six major steps:

In step 4.1, "Insertion request", a trusted device 2 of a community 5 named *Detector* detects an insertion request from a new device 1 named *New*.

In step 4.2, "Forward request", this request is forwarded to a predetermined controller 4 of the community 5.

In step 4.3, "User request", the controller 4 warns the user 3 upon receiving a request from the Detector device 2.

In step 4.4, "User confirm", the user 3 explicitly authorizes the insertion of the new device 1 in the community 5.

In step 4.5, "Forward confirm", the confirmation is forwarded backward to the Detector device 2 and then, in step 4.6, "Insertion confirm", the insertion is confirmed to the new device 1. It is to be noted that steps 4.5 is not mandatory; in a variant, the controller 4 may directly confirm insertion to the new device 1.

These six main steps can be secured by the use of some keys or other cryptographic material. They can be followed by other optional steps to exchange additional key material or protocol related information.

But this method, as the previous ones does not allow the user 3 to choose the device used to authorize the insertion.

In view of the methods described above, we can see that no known prior art method lets the user choose the device that he wants to use for authorizing insertion of a new device. Either the detector device or a predetermined controller device must be used.

On aim of the invention is therefore to let the user choose a device to validate the insertion of a new device. It should be possible that the chosen device is not the one that detected the arrival of the new device. Moreover, it should be possible that the chosen device cannot directly communicate with the new device. In such cases, the following technical problems may arise:

- How will the chosen device be informed that a new device demands insertion into the community? How will it be informed of the identity of the new device?

- Conversely, how will the new device be informed that the user had chosen a device to validate the insertion? How will the new device learn the identity of the chosen device?
- How will necessary information messages flow between these two devices?

In addition, even if we solve these technical problems, both the new device and the device chosen by the user to validate the insertion of the new device must be given the possibility play the rest of their usual insertion protocol, as if they were under standard insertion conditions. In other words, the gain of user friendliness must not result in heavy protocol rewriting that would render invention incompatible with state of the art solutions already deployed.

SUMMARY OF THE INVENTION

In order to overcome the above-mentioned problems, the invention proposes a method for inserting a new device (*x*) in a community of devices wherein each device of the community stores insertion requests received from new devices and forwards these insertion requests to a device (*b*) chosen by a user of the community for confirming authorization to join the community.

BRIEF DESCRIPTION OF THE DRAWINGS

The various features and advantages of the present invention and its preferred embodiments will now be described with reference to the accompanying drawings which are intended to illustrate and not to limit the scope of the present invention and in which:

Fig. 1a, 1b and 1c, already described, illustrate transitiveness of mutual trust relations in communities.

Fig. 2, 3 and 4, already described, illustrate known examples of authorization of a new device in a community using a trusted device (Fig. 2) or a predetermined controller (direct method – Fig. 3 or indirect method – Fig. 4).

Fig. 5 to 7 illustrate a process according to the invention.

Fig. 8 is an example of device selection interface.

Fig. 9 to 11 illustrate three stages of a preferred embodiment of the process of the invention.

Fig. 12 shows network messages exchanged between devices and user interactions during the insertion process.

Fig. 13 illustrates a situation in which the invention is particularly useful.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

One idea of the invention is to de-correlate the detection of a new device and the choice of a specific device of the community for confirmation. A device will inquire for pending insertion requests only once it had been chosen by the user.

Fig. 5 illustrates the two first steps of a new insertion process according to the invention.

In step 5.1 an insertion request "insreq(id_x)" is sent by a new device x and is detected by a device *a* of a community 5. At this step, device *a* only stores the request for further use. Then, in step 5.2, the user 3 chooses any device in the community 5 and inquires for pending insertion requests. The chosen device may be device *a* or another device (in Fig. 5 for example, the user has chosen another device, called *b*).

As the first step 5.1 is triggered by the incoming of a new device *x*, and the second step 5.2 is triggered by a user action, there may be an arbitrarily long delay between the two steps. As device *b* had been chosen freely, there is no *a priori* reason that *b* is aware of the existence of an insertion request 5.1 from device *x*.

The two next steps of the insertion process according to invention allow device *b* to collect pending insertion request. These steps are shown in Fig. 6.

In step 6.3, the device *b* (chosen by the user 3) asks all reachable community devices for pending insertion requests. In step 6.4, all community devices that stored insertion requests answer to *b* (here, device *a* answers).

It is to be noted that once community devices have answered to device *b*, they immediately forget pending insertions requests. If they didn't forget this information, they would pointlessly answer to further requests from device *b*.

At the end of step 6.4, device *b* is informed that a device *x* exists and required insertion. Device *b* might have received other insertion requests from candidates *y*, *z*...

The two last steps of the process according to the invention are meant to inform device *x* that it will be inserted by device *b*. They are shown in Fig. 7.

In step 7.5, the user 3 chooses, among all pending insertion requests, which one must be satisfied. In Fig. 7, the request from device *x* will

be satisfied. For secure insertion process, this step is also used to validate that the identity claimed by *x* is correct. Then, in step 7.6, the device *x* is informed that device *b* was chosen by the user. It is to be noted that device *b* is informed of a mean of communication with device *x*: here, through device *a* that relayed the insertion request. This specific aspect is not part of the invention, but uses state of the art routing methods.

At the end of these 6 steps illustrated by Fig. 5 to 7, the following situation holds: one device (*b*) has been chosen by the user 3 for insertion of one new device (*x*), and both devices are informed of the existence and status of the other. This is the required situation for any state of the art insertion process to begin.

In order to be able to implement the invention, the devices should have the following capacities. They should be able:

- To store a set of pending insertion requests. To that end, devices will use a memory called "lastreq", that can store at least one pending insertion request. A storage time limit (for instance: one day) is possible but not required for the invention;
- To forget pending insertion requests: the lastreq memory can be erased;
- To access at least one network protocol with broadcast capacity. Broadcast provides a way to address all other devices with one single network transmission and is present in most protocols used by communities;
- To store the answer when asking for pending requests. Device will use a chained list of device identifiers. This chained list is denoted "Pending_List";
- To allow the user to choose among several devices that required insertion: the device will provide a simple user interface, adapted to its display capacities An example of a screen interface is given in Fig. 8 where device identifiers are defined by hexadecimal numbers.

We will now describe in more details a preferred embodiment of the invention. This description will use notations that are used and explained in more details in document "*Secure Long Term Communities in Ad Hoc Networks, N. Prigent, C. Bidan, J.P. Andreaux, O. Heen, 1st ACM Workshop on*

Security of Ad Hoc and Sensor Networks (SASN)" previously mentioned. For example the device identifier of a device *x* is denoted id_x.

In the following description, *x* is the new device, *a* is one of the detectors, *b* is the user chosen device. The algorithm performed by devices to realize the invention contains three parts, described hereafter.

First stage of the algorithm

This stage allows emission, reception and storage of insertion requests. Its beginning corresponds to step 5.1 in Fig. 5. The user is not involved at this stage.

Fig. 9 illustrates actions performed in device *a*. All other devices behave similarly at this stage. Steps 501 to 510 of this first stage of the process are explained bellow.

501: device *a* is idle; network is idle. Execution is transferred to 502.

502: device *a* broadcasts insertion request. Execution is transferred to 503.

503: device *a* waits for network messages. If no message arrives after a timeout, execution is transferred back to 502. In an exemplary embodiment the timeout value is set to 10 seconds.

504: Device *a* just received an insertion request "insreq(id_x)" sent by device *x*. Execution is transferred to 505.

505: if *x* from "insreq(id_x) is already a trusted device for *a*, execution is transferred back to 502. Otherwise, execution is transferred to 506. Note that the above-mentioned document "*Secure Long Term Communities in Ad Hoc Networks*" indicates a simple way for a device *a* to check if a device *x* is trusted or not.

506: the insertion request from device *x* is stored in the "lastreq" memory of device *a*. Note that the former content of "lastreq" is lost. Other embodiments may manage more complex memories, such as FIFO or arrays.

507: device *a* just received a "seek_pendings" message. Execution is transferred to 508.

508: if the "lastreq" memory of device *a* is empty, device *a* does not answer to the "seek_pendings", and execution is transferred back to 502.

509: device *a* answers with a message "is_pending(lastreq)". Execution is transferred back to 502.

Note that steps 507 to 509 corresponds to steps 6.3 and 6.4 illustrated by Fig. 6.

510: The "lastreq" memory of device *a* is emptied. Note that this step is not mandatory, but prevents from further unnecessary message exchanges.

Second stage of the algorithm

This stage first lets the user choose the device (denoted *b*) that he will use to authorize insertion. Then device *b* demands and obtains pending insertion requests from other devices. User needs appropriate credentials to log on device *b*. This stage of the process corresponds to step 5.2 in Fig. 5 and steps 6.3 and 6.4 in Fig. 6.

Fig. 10 shows the behavior of a device *b* at this stage. Steps 601 to 607 of this second stage of the process are explained bellow.

601: A user with appropriate credentials decides to log on device *b*. Execution is transferred to 602.

602: if user selects the command "start insert!", then execution is transferred to 604. Otherwise, it is transferred to 603.

603: User makes normal use of the device *b* then logs off. Execution is transferred to 601.

604: "Pending_list" is set to the value of "lastreq". This step is useful for treating an insertion request that could have been received by device *b* himself. Execution is transferred to 605.

605: the message "seek_pendings" is broadcasted in the community. Execution is transferred to 606.

606: device *b* is awaiting answers from other community devices. If an answer is received, execution is transferred to 607. After a predetermined timeout, execution is transferred to the next stage of the algorithm illustrated in Fig. 11.

607: an answer is_pending(id_x) was received. Then id_x is added to "Pending_list".

Note that the size of "Pending_list" is not infinite. The described embodiment works when the size is one: new requests overwrite older requests. However, a list size of 16 elements for instance would avoid too many retries and unnecessary message transmissions.

Third stage of the algorithm

This stage lets the user choose one pending request and satisfy it. This stage of the process must happen both on device *b* (the user chosen device) and on device *x* (the new device). The order has no importance.

This stage corresponds to steps 7.5 and 7.6 of Fig.7. At the end of this part, the standard insertion process described in document "*Secure Long Term Communities in Ad Hoc Networks*" continues normally.

Fig. 11 shows the execution steps for this stage, when executed by device *b*. Steps 701 to 703 of this second stage of the process are explained bellow.

701: the user selects one pending insertion request from "Pending_list" of device *b*. In the case described here, there exists at least one insertion request, emitted by device *x*. Execution is transferred to 702.

702: the list of unilaterally trusted devices (UT) is completed with device *x*. This corresponds to the first step of normal insertion process described in the above-mentioned document. Execution is transferred to 703.

703: an insertion request is transmitted by device *b* to device *x*. This point is important as, until now, device *x* is not aware of device *b* existence nor identity. Note that, when executed by device *x*, this step results in the transmission of an insertion request to device *b*. This transmission is not strictly necessary but does not disturb the whole process.

At the end of these three stages of the process, all necessary conditions for starting standard insertion protocol from the above-mentioned document are fulfilled. These conditions are recalled here:

- Device *b* knows devices *x* identity, Id_x.
- Device *b* had been authorized by the user to insert device *x*.
- Device *x* knows devices *b* identity, Id_b.
- Device *x* has been authorized by the user to insert device *b*.

In the preferred embodiment of the invention, device identifier length is set to 128 bits. The size of "lastreq" memory is set to 128 bit also, that is "lastreq" can store on pending request at a time. Maximum length of "Pending_List" is set to 3 items.

Fig. 12 illustrates the sequence of exchanged messages during the insertion of a new device *x*, detected by a device *a*, with device *b* as the user chosen device. All these devices implement the three stages of the process that have been described above.

User actions regarding devices *b* and *x* are also represented. Dotted horizontal lines represents arbitrary delays before user actions. These delays

do not disturb the process: user is not obliged to respect any predetermined delay between actions.

Reference numbers 12-1 to 12-6 that appear on certain messages refer to labels N-1 to N-6 from Fig. 5, 6 and 7 with "N" representing the number of the figure. Thick rectangles marked "start insert with *x*" and "start insert with *b*" mean that concerned devices may start their standard insertion protocol.

The invention presents the following advantages.

The invention facilitates device insertion into a community. It brings more flexibility to the user when he decides to insert a device in a community and respects the constraint that existing protocol are not modified.

The invention lets the user choose the best-suited device for new device insertion. For example, when device insertion needs visual interface with the user, the user will prefer a device with good screen capabilities.

In the case of a community used by several users, one user may not be able to log on all devices of the community. Using prior art methods, this user can insert new devices only if he is granted log on credentials to the predetermined controller or the detector. Using the invention, this restriction does not apply anymore: the user is always able to insert a new device in the community.

There is an advantage in the case illustrated by Fig. 13: when the community crosses (at least) one open network, such as the Internet. For instance, this corresponds to the case of a home network that spreads in user home and in users secondary residence, both sites being connected using ADSL Internet accesses. In this case, one user called "User 2" in Fig. 13 brings a new device *x* for insertion in the community. It may happen that "User 2" is not allowed to log on any community device in its immediate environment. In this case, prior art method do not allow insertion, and "User 2" would be obliged to obtain login credentials. Using the invention, "User 2" may get help from remote "User 1" who will choose a device *b* and start the insertion process. Note that "User 1" needs nothing more than credentials to log on device *b*. This allows device *x* insertion with no credentials revealed to "User 2".

CLAIMS

1. Method for inserting a new device (*x*) in a community of devices wherein each device of the community stores insertion requests received from new devices and forwards these insertion requests to a device (*b*) chosen by a user (3) of the community for confirming authorization to join the community.

ABSTRACT

Method for inserting a new device in a community of devices

5

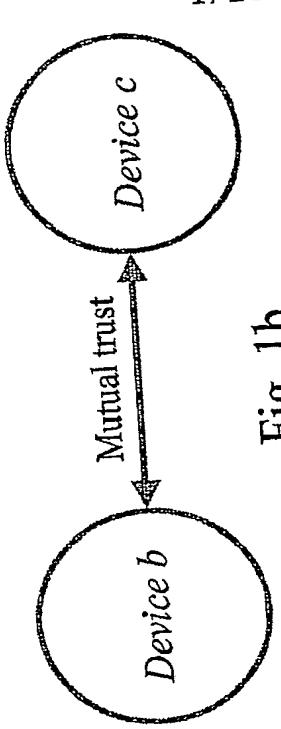

Fig. 1b

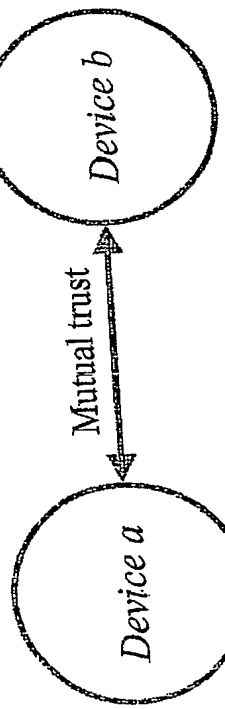

Fig. 1a

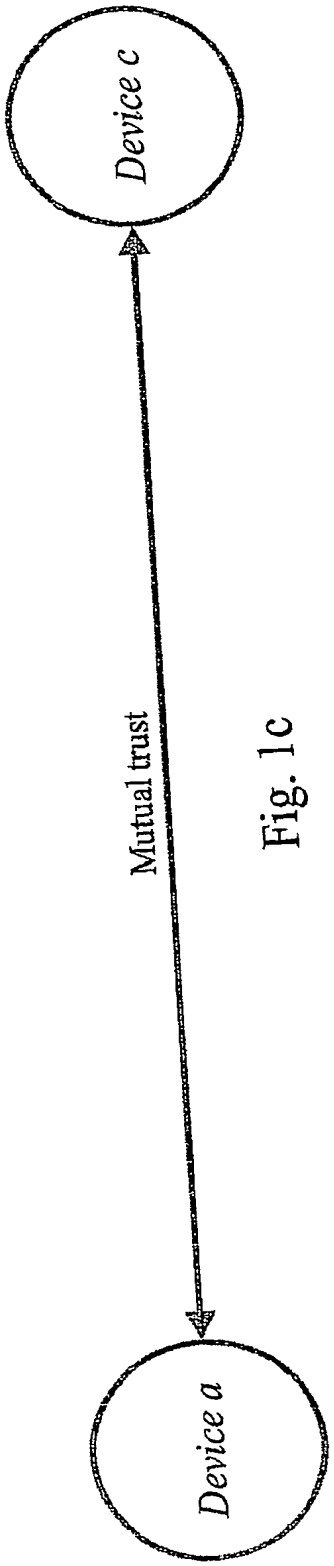

Fig. 1c

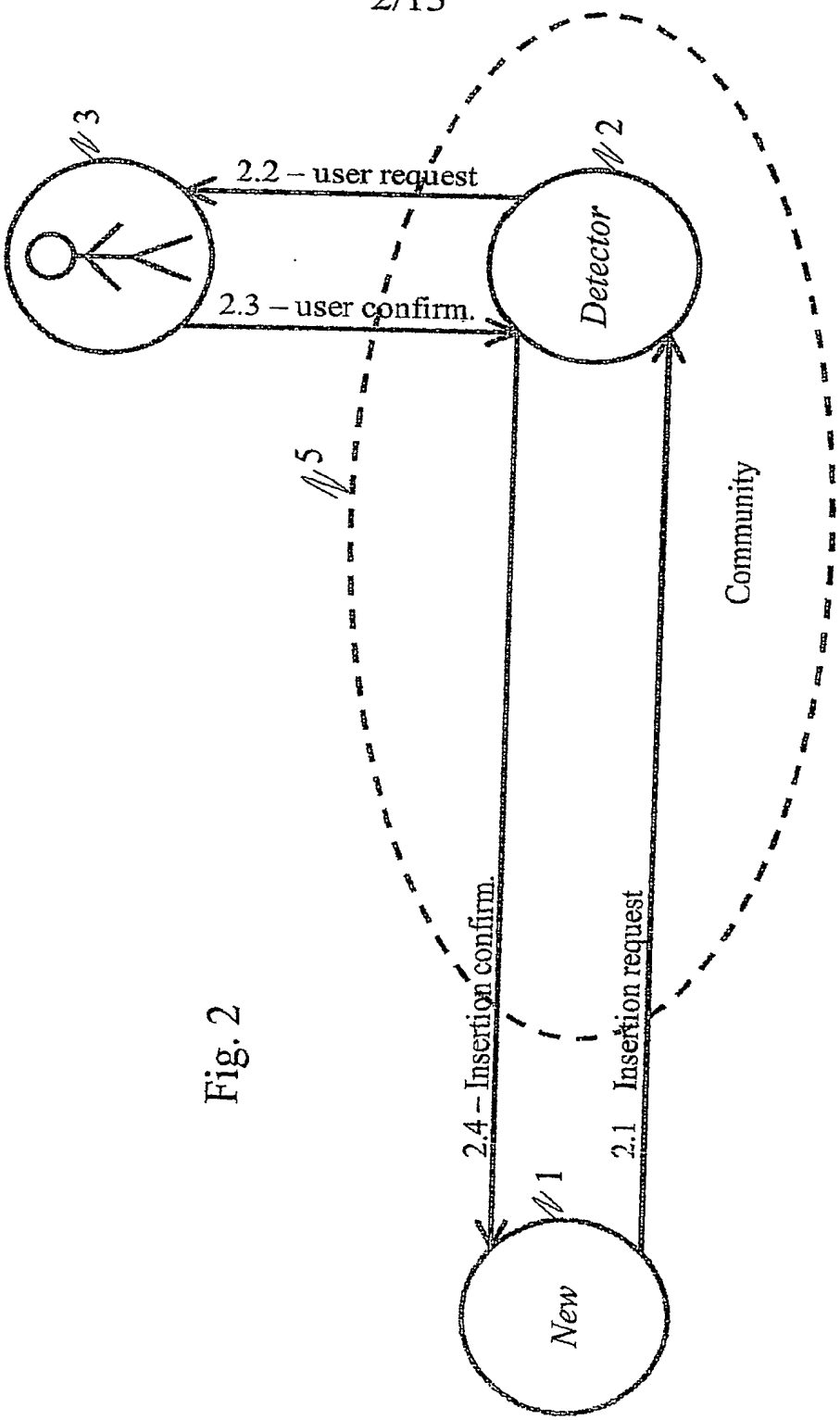
3
2
2.2 – user request
2.3 – user confirm.
Detector
5
Community
2.4 – Insertion confirm.
2.1 Insertion request
1
New

Fig. 2

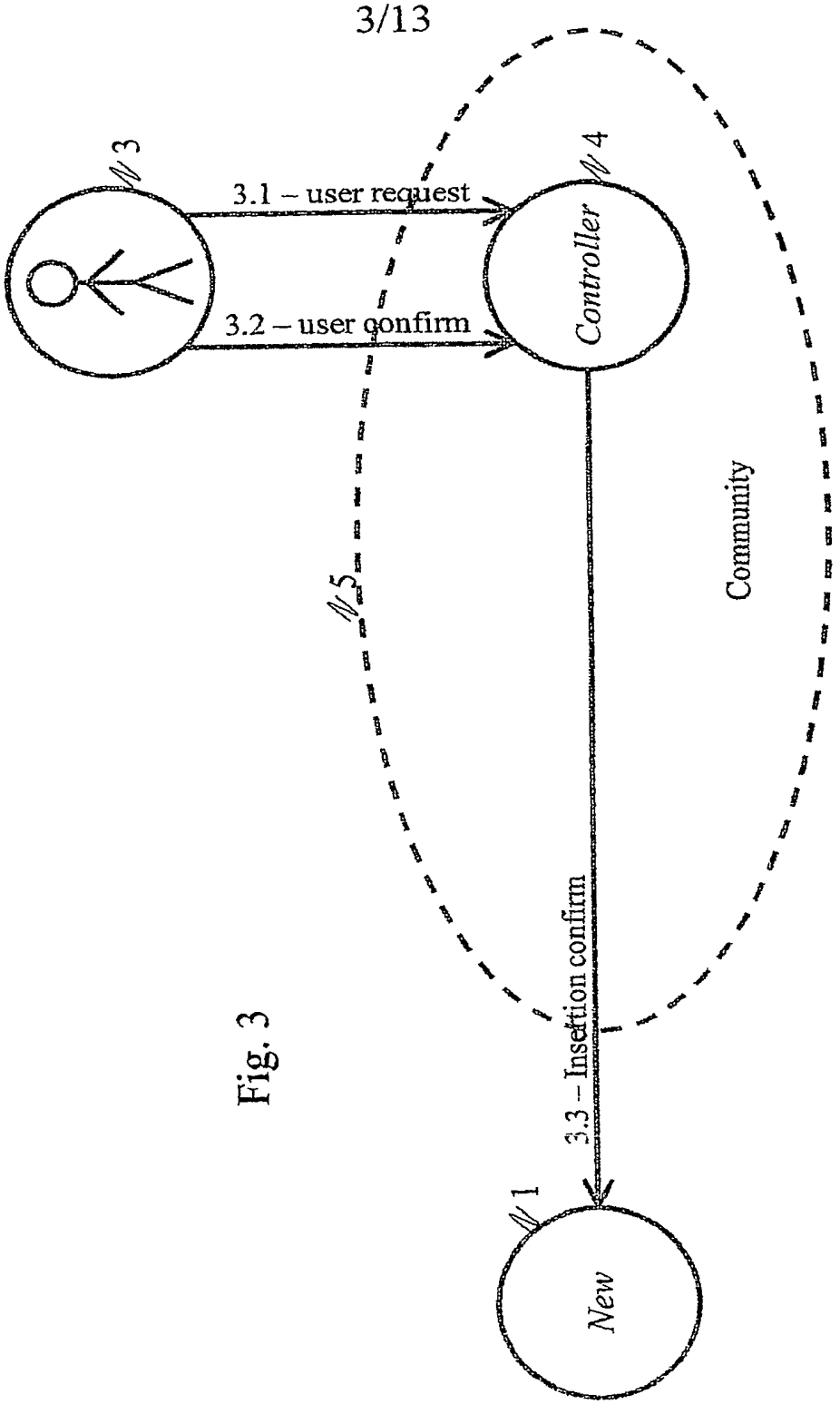
3
4
3.1 – user request
3.2 – user confirm
Controller
5
Community
3.3 – Insertion confirm
1
New

Fig. 3

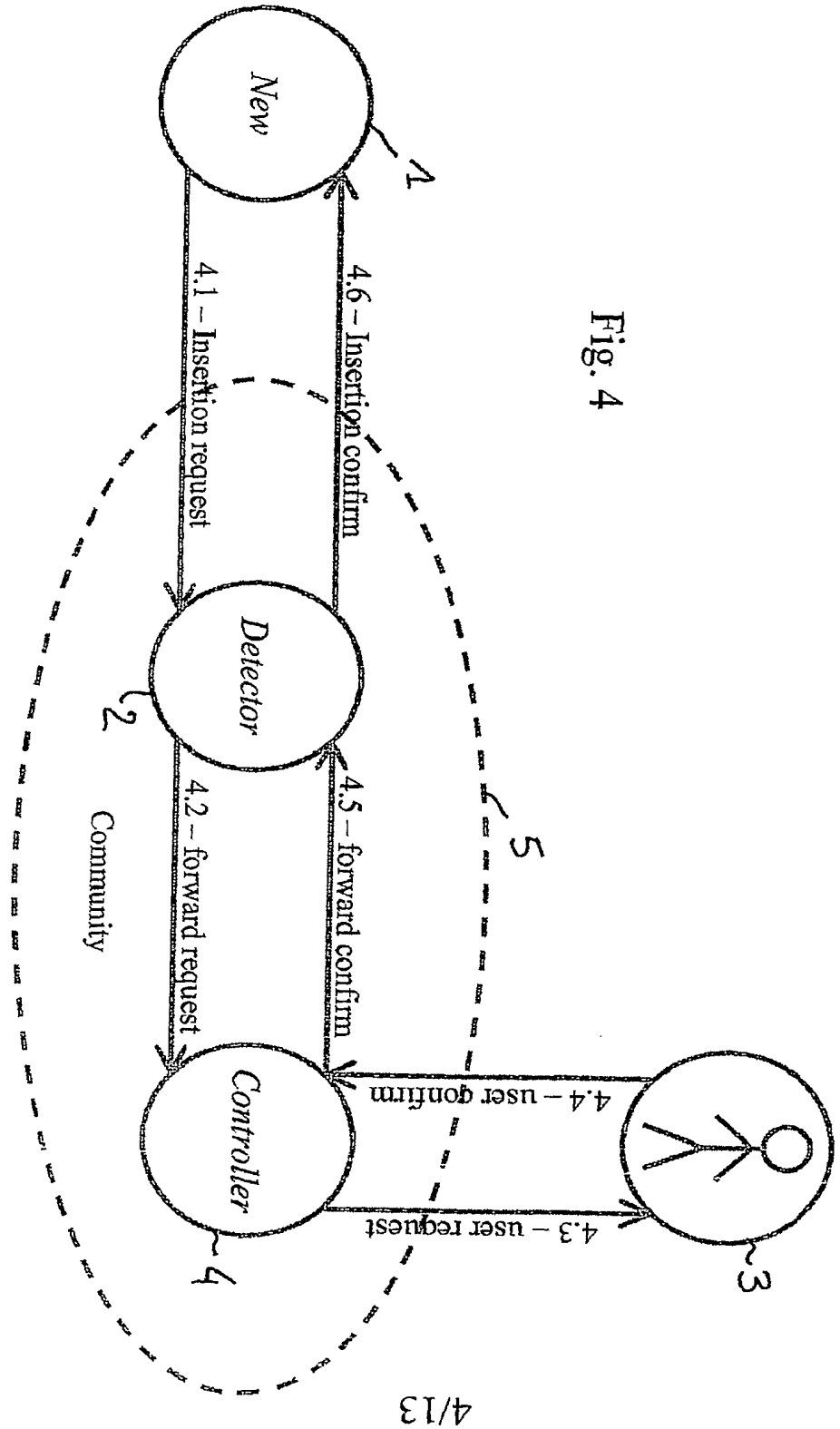
New
1
4.6 – Insertion confirm
4.1 – Insertion request
Detector
2
Community
5
4.2 – forward request
4.5 – forward confirm
Controller
4
4.4 – user confirm
4.3 – user request
3

Fig. 4

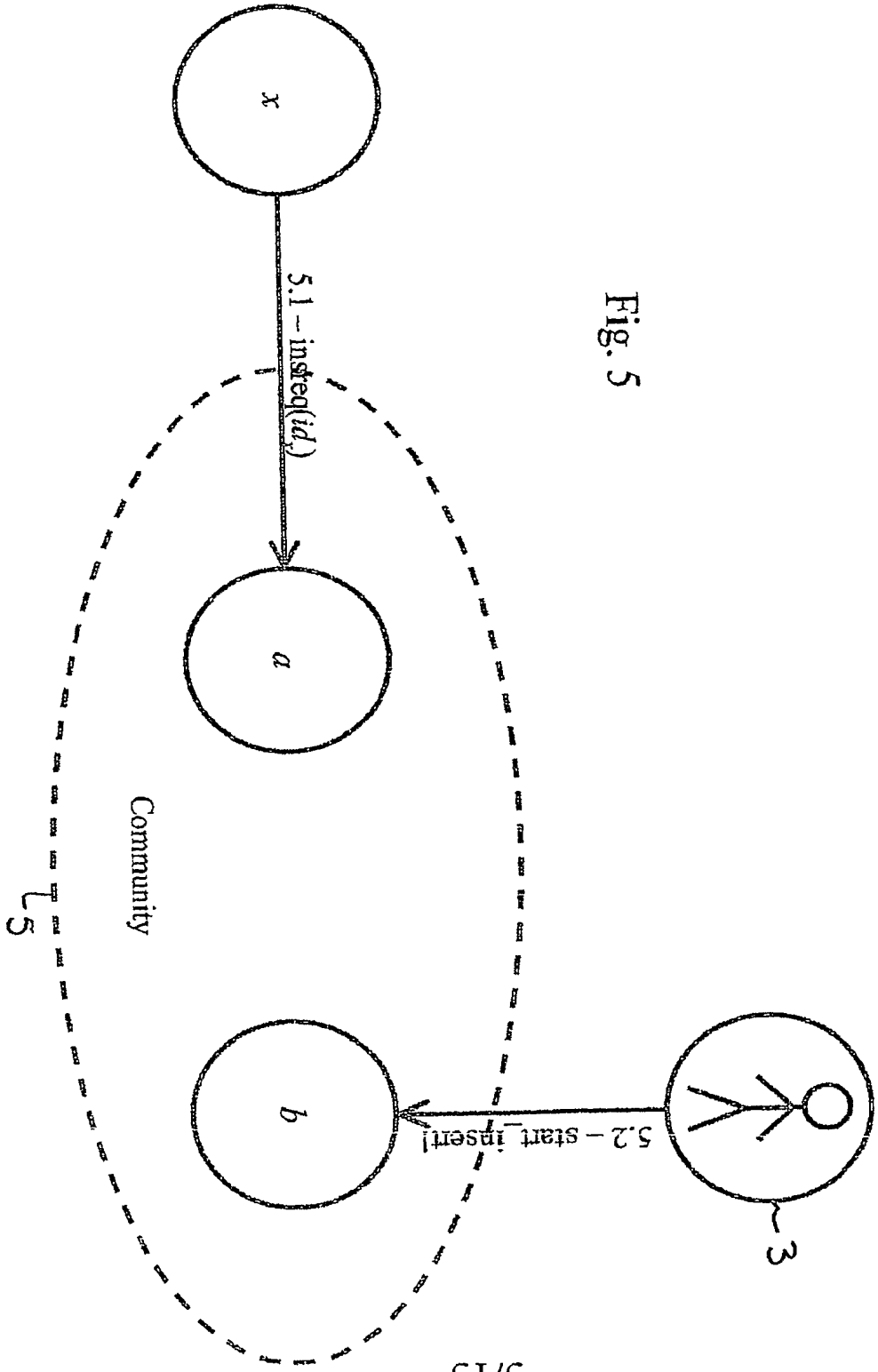

Fig. 5

Fig. 6

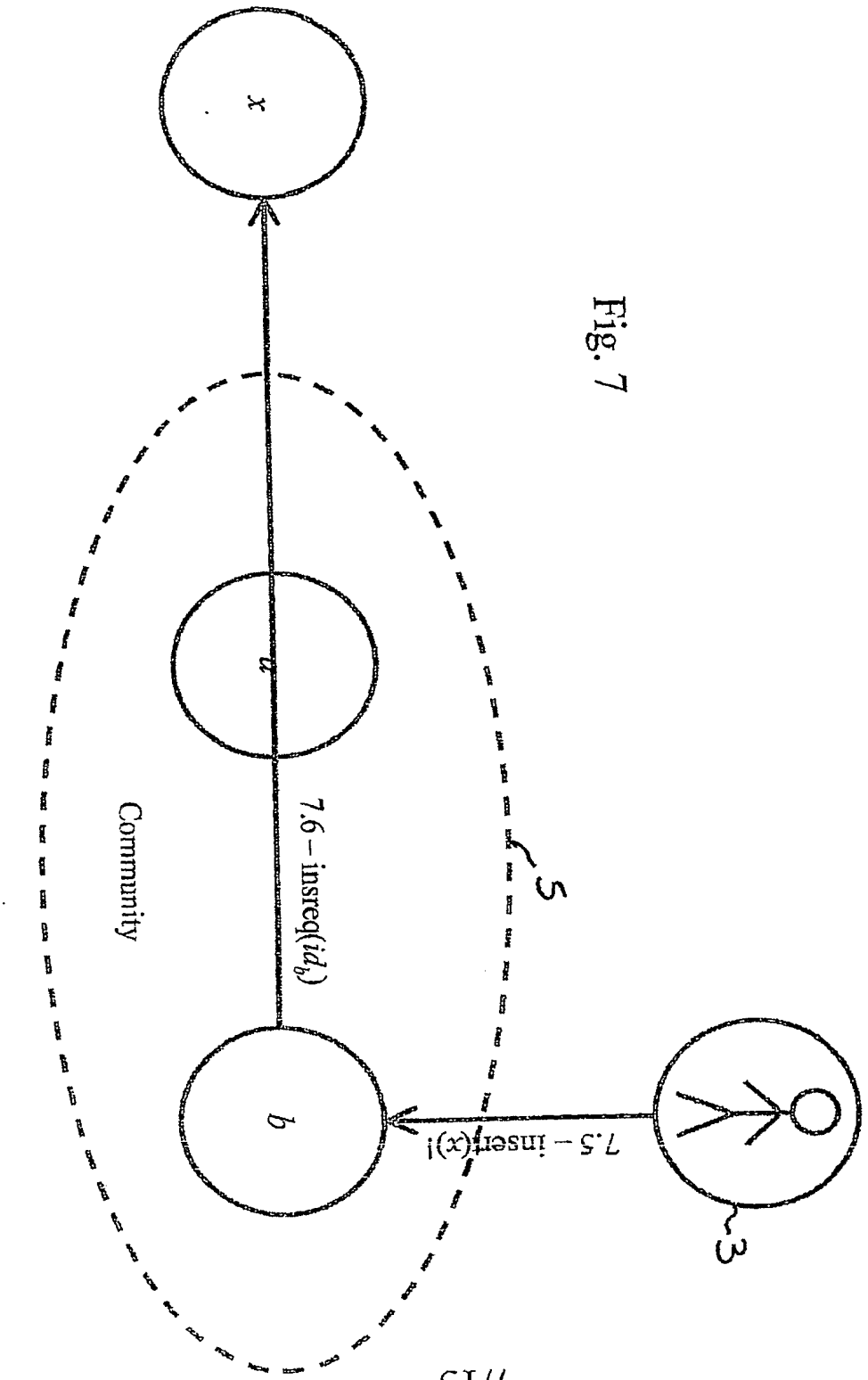

Fig. 7

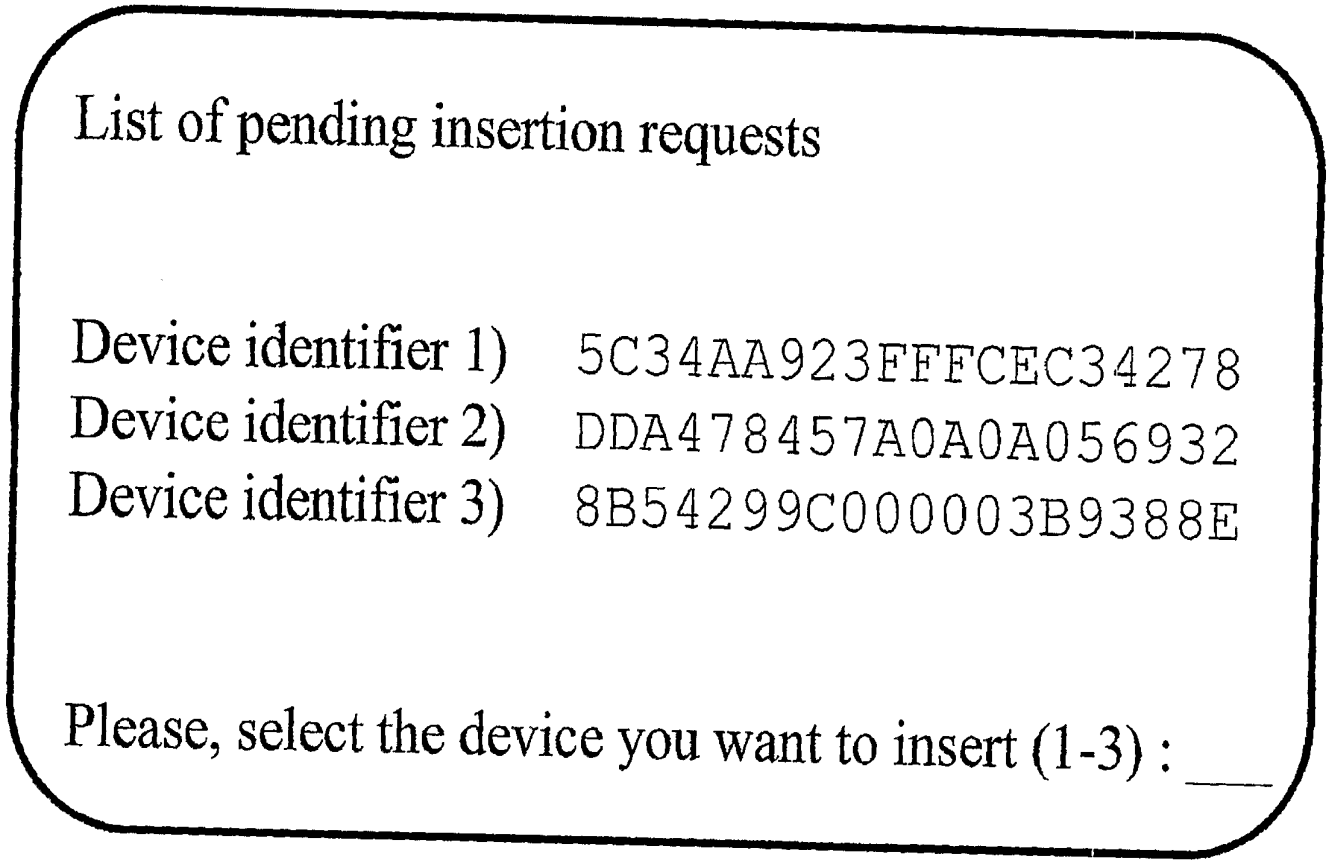

Fig. 8

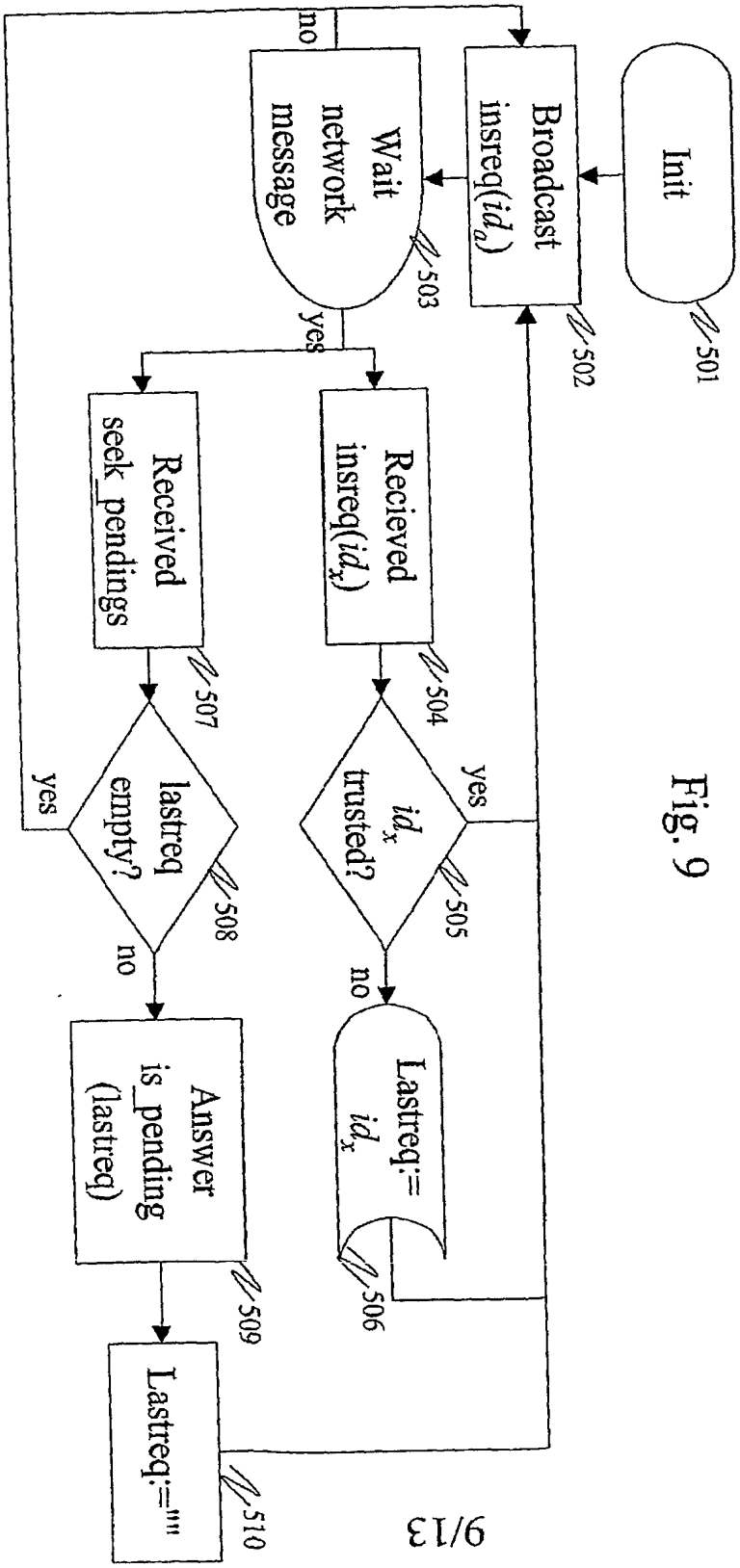
Init
501
Broadcast insreq(id_a)
502
Wait network message
503
no
yes
Recieved insreq(id_x)
504
id_x trusted?
505
yes
no
Lastreq:= id_x
506
Received seek_pendings
507
lastreq empty?
508
yes
no
Answer is_pending (lastreq)
509
Lastreq:=""
510

Fig. 9

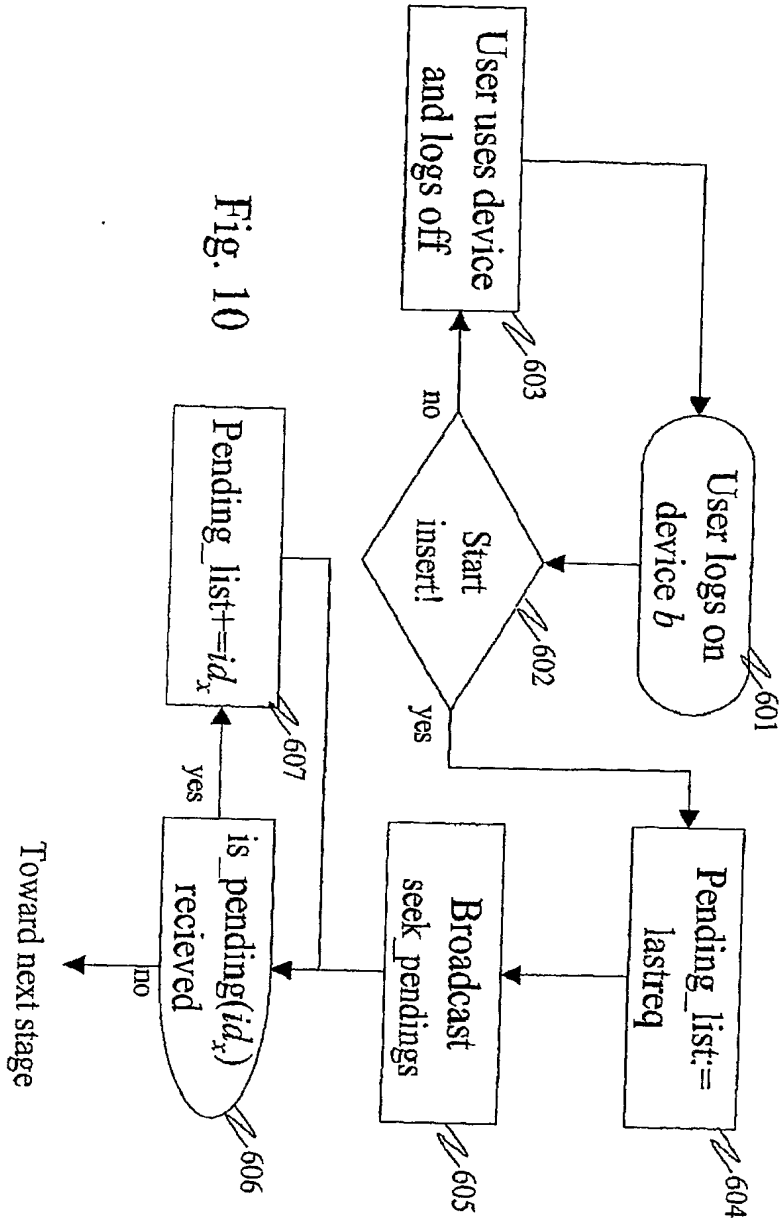
User logs on device b
601
Start insert!
602
no
yes
User uses device and logs off
603
Pending_list:= lastreq
604
Broadcast seek_pendings
605
is_pending(id_x) recieved
606
no
yes
Pending_list+=id_x
607
Toward next stage

Fig. 10

User selects command Insert(x)!
701
UT+=x
702
Send inserq(id_b) to x
703
Toward standard insertion protocol between b and x

Fig. 11

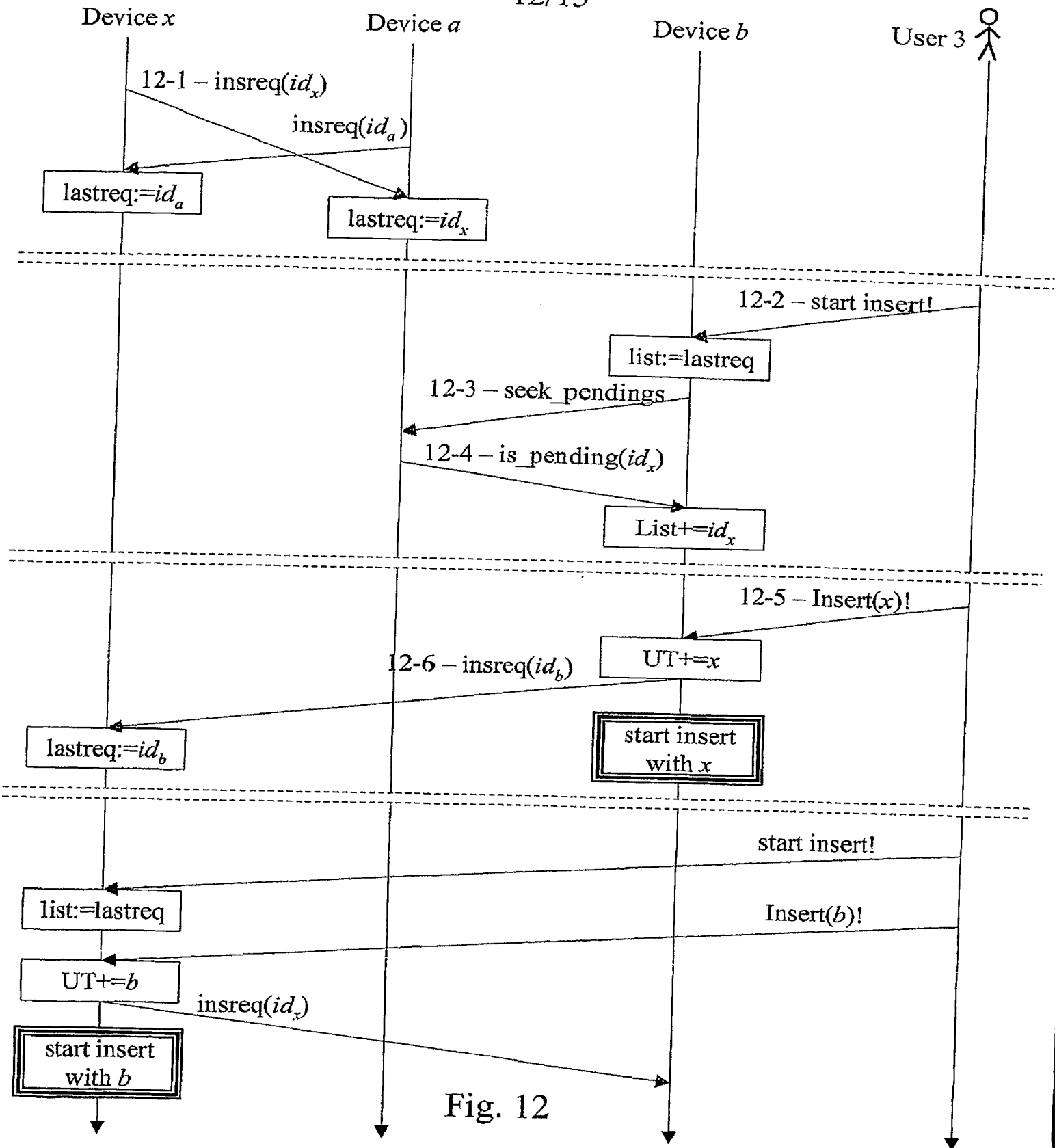
Device x
Device a
Device b
User 3
12-1 – insreq(id_x)
insreq(id_a)
lastreq:=id_a
lastreq:=id_x
12-2 – start insert!
list:=lastreq
12-3 – seek_pendings
12-4 – is_pending(id_x)
List+=id_x
12-5 – Insert(x)!
UT+=x
12-6 – insreq(id_b)
lastreq:=id_b
start insert with x
start insert!
list:=lastreq
Insert(b)!
UT+=b
insreq(id_x)
start insert with b

Fig. 12

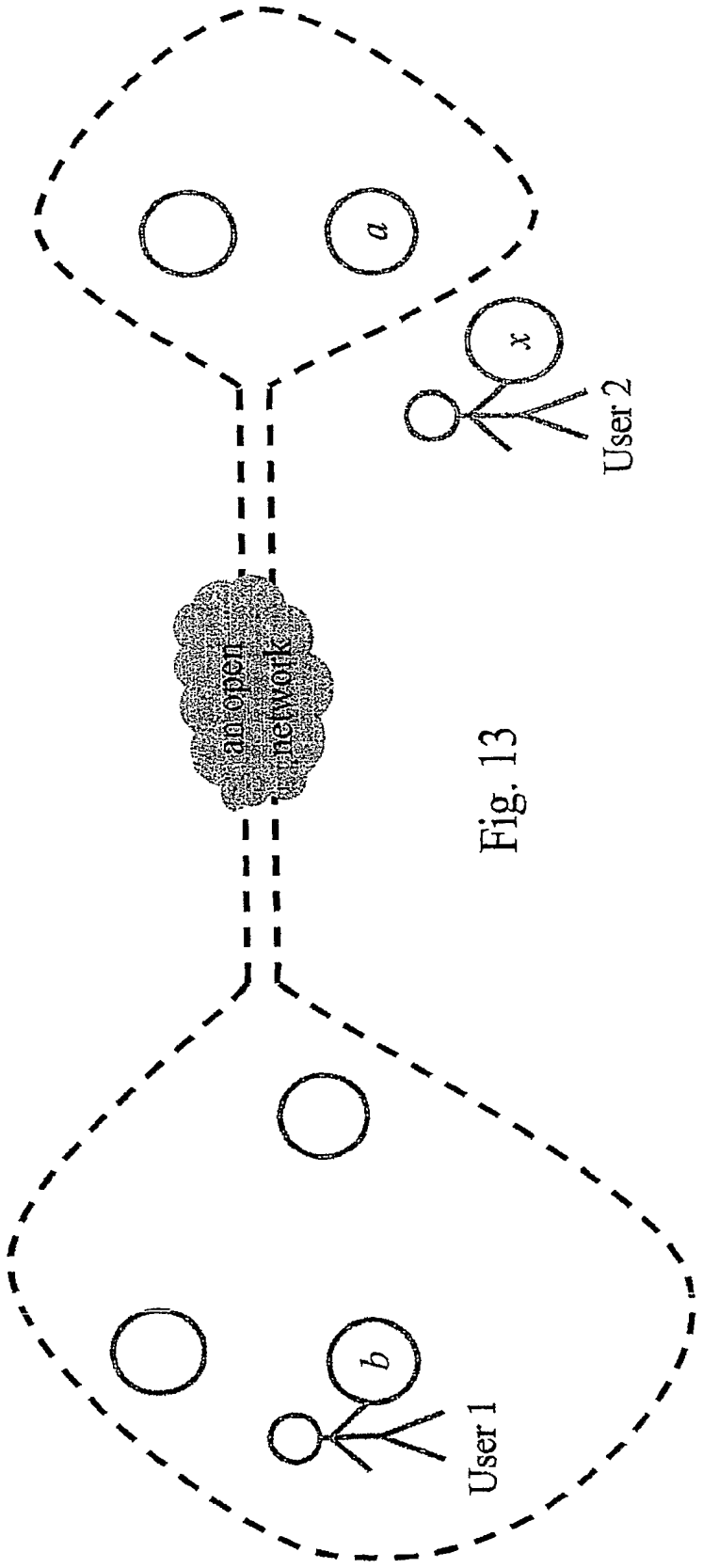

Fig. 13